Fast Yet Rich Morphological Analysis

Mohamed Altantawy, Nizar Habash, and Owen Rambow
Center for Computational Learning Systems
Columbia University
New York, NY, USA
{mtantawy,habash,rambow}@ccls.columbia.edu

Abstract

Implementations of models of morphologically rich languages such as Arabic typically achieve speed and small memory footprint at the cost of abandoning linguistically abstract and elegant representations. We present a solution to modeling rich morphologies that is both fast and based on linguistically rich representations. In our approach, we convert a linguistically complex and abstract implementation of Arabic verbs in finite-state machinery into a simple precompiled tabular representation.

1 Introduction

Arabic is a morphologically rich and complex language, characterized by a combination of templatic and affixational morphemes, complex morphological, phonological and orthographic rules, and a rich feature system. Arabic morphological analysis and generation are important to many NLP applications such as machine translation (Habash, 2007; Kholy and Habash, 2010) and information retrieval (Aljlayl and Frieder, 2002). Much work has been done on Arabic morphological analysis and generation in a variety of approaches and at different degrees of linguistic depths.

There is a continuum of approaches which is characterized by its two poles: on one end, very abstract and linguistically rich representations and rules (often based on particular theories of morphology) are used to derive surface forms; while on the other end, simple and shallow techniques focus on efficient search in a space of precompiled (tabulated) solutions. The first type is typically implemented using finite-state technology and can be at many different degrees of sophistication and detail. An example of this type of implementation is the MAGEAD (Morphological Analysis and GEneration for Arabic and its Dialects) system (Habash et al., 2005; Habash and Rambow, 2006). This system, which we use as starting point in this paper, compiles abstract high-level linguistic information of different types to finite state machinery. The second type is typically not implemented in finite-state technology. Examples include the Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter, 2004) and its extension ALMORGEANA (Habash, 2007). These systems do not represent the morphemic, phonological and orthographic rules directly at all, and instead compile their effect into the lexicon itself.

Numerous intermediate points exist between these two extremes (e.g., (Smrž, 2007)). The various approaches typically trade off different degrees of speed and memory (model size) for more abstract and elegant representations. Precompiled tabular systems usually have fast response time when implemented using hash tables. The cost of building the linguistic resources manually is the main drawback for this approach since such resources are prone to error and hard to debug and extend. Linguistically sophisticated systems are easier to understand and extend and they allow for modeling morphology without lexicons (to address unknown forms); however, with the complexity of some languages' morphology, the finite-state transducers (FSTs) tend to become extremely large, causing a significant deterioration in response time.

In this paper, we present a solution to the modeling of rich morphologies that combines the best of these two approaches. In previous work, we presented MAGEAD, a multi-tier finite-state implementation of Arabic morphology (Habash et al., 2005; Habash and Rambow, 2006; Altantawy et al., 2010). We improve the speed by automatically converting our FST-based MAGEAD system to a precompiled tabular implementation that preserves all of the rich linguistic information used in MAGEAD's design. The new system, MAGEAD-EXPRESS is not only much faster and smaller in size, but it also still allows linguistically based abstract changes and updates in its model. Furthermore, MAGEAD-EXPRESS produces complete linguistic analyses that include intermediate levels of representation, an advantage MAGEAD does not have readily in its output. The only disadvantage of MAGEAD-EXPRESS is its inability to produce analyses for unknown words (unlike MAGEAD).

The paper is organized as follows. We give a short introduction to Arabic morphology in section 2. The related work is discussed in section 3. We then present MAGEAD and MAGEAD-EXPRESS in sections 4 and 5 respectively. The extraction process of MAGEAD's linguistic information is discussed in section 6. Finally, the evaluation is presented in section 7.

2 Arabic Morphology

For an extensive discussion of Arabic morphology from a computational perspective, see (Habash, 2010); we give a short overview here. Arabic has a rich and complex morphology. This is due to its numerous linguistic features, such as gender, number, mood and case, and the existence of two types of morphemes: templatic and affixational (concatenative). **Templatic morphemes** come in three types that are equally needed to create a word stem: roots, patterns and vocalisms. The **root morpheme** is a sequence of typically three consonants (termed *radicals*) that signifies some abstract meaning shared by all its derivations. For example, the words كَتَبَ *kataba*[1] 'he wrote' and كَاتِب *kaAtib* 'writer' share the root morpheme (ك-ت-ب) *ktb* 'writing-related'. The **pattern morpheme** is an abstract template in which roots and vocalisms are inserted. The **vocalism morpheme** specifies which short vowels to use with a pattern. We represent the pattern as a string made up of numbers to indicate radical position, of the symbol *V* to indicate the position of the vocalism, and of pattern consonants (if needed) following (Habash et al., 2005). As an example, the word stem كَتَب *katab* is constructed from the root (ك-ت-ب) *ktb*, the pattern *1V2V3* and the vocalism *aa*. This is represented as ⟨ 1V2V3, ktb, aa ⟩. Arabic **affixes** can be **prefixes** such as (+سَ) *sa+* 'will', **suffixes** such as (هُم+) *+hum* 'they [masculine]' or **circumfixes** such as (تَ+ +نَ) *ta++na* '[imperfective subject 2nd person feminine plural]'. We do not distinguish between clitics and inflectional affixes in this paper. Multiple affixes can appear in a word. An Arabic word is constructed by first creating a word **stem** from templatic morphemes, then adding affixational morphemes. The process of combining morphemes involves a number of morphemic, phonological and orthographic rules that modify the form of the created word so it is not a simple interleaving or concatenation of its morphemic components. See example in section 4.

As in other languages, surface word forms in Arabic that differ only in inflectional morphology can be grouped into a lexicographic abstraction called a *lexeme*. A lexeme is usually represented by a conventionally chosen word form called the *lemma*, but it can also be represented by a stem and an inflectional class, and in Arabic (as in other Semitic languages), we can represent it as the set of radicals and an inflectional class. Thus, we can represent a morphological analysis of any Arabic word form in terms of its lexeme (i.e., radicals and inflectional class), and a set of feature-value pairs which the morphology of the word form represents. For example, the MSA verb وَلِيَدعُوهُ *waliyadςuwhu* 'and that they invite him' corresponds to the following lexeme-and-features representation:

[1]Arabic transliteration is in the HSB scheme (Habash et al., 2007): (alphabetically) *AbtθjHxdðrzsš SDTĎςγfqklmnhwy* and (additional symbols) ' ء, Â أ, Ǎ إ, Ā آ, ŵ ؤ, ŷ ئ, ħ ة, ý ى.

(1) [1d][2ς][3w][mbc:verb-I-au-tr]
[asp:I][vox:act][mod:s][per:3][gen:m][num:p]
[cnj:w][prt:l][pro:3MS]

This representation indicates (in order of symbols above) that the root radicals of the word are d-ς-w; that the verb belongs to a particular inflectional class called *I-au* and that it is transitive; that aspect is indicative; voice, active; mood, subjunctive; person, third; gender, masculine; number, plural; that it has the conjunction proclitic *w* 'and'; that it has particle proclitic *l* 'for/that'; and that it has a pronominal enclitic that is 3rd person masculine singular.

3 Related Work

There has been a considerable amount of work on Arabic morphological analysis; for an overview, see (Al-Sughaiyer and Al-Kharashi, 2004). The first large-scale implementation of Arabic morphology within the constraints of finite-state methods is that of Beesley et al. (1989) (the "Xerox system") with a 'detouring' mechanism for access to multiple lexica, which gives rise to other works by Beesley (1998) and Beesley and Karttunen (2000) and, independently, by Buckwalter (2004). Unlike the Xerox system, the Buckwalter Arabic Morphological Analyzer (BAMA) uses a hard-coded tabular approach with a focus on analysis into surface morphemes (discussed above). Buckwalter's work has been since extended to handle generation as well as the lexeme-and-features representation (ALMORGEANA, (Habash, 2007)) and functional morphology in Arabic (ELIXIR, (Smrž, 2007)).

Finite-state handling of templatic morphology has been demonstrated using a variety of techniques for several Semitic languages other than Arabic including Akkadian (Kataja and Koskenniemi, 1988), Syriac (Kiraz, 2000), Hebrew (Yona and Wintner, 2005), Amharic (Amsalu and Gibbon, 2005), and Tigrinya (Gasser, 2009). Kay (1987) proposes a framework for handling templatic morphology in which each templatic morpheme is assigned a tape in a multi-tape finite state machine, with an additional tape for the surface form. Kiraz (2000) extends Kay's approach and implements a multi-tape system for Modern Standard Arabic (MSA) and Syriac. The MAGEAD system (Habash et al., 2005; Habash and Rambow, 2006) extended Kiraz's work through a new implementation using AT&T finite state machine toolkit (Mohri et al., 2000) with an eye on handling morphology for Arabic and its dialects. In this work, we start with MAGEAD and address three of its weaknesses: its slowness, its size, and the absence of rich morphological information in its output despite of its presence in model specifications.

4 The MAGEAD System

4.1 MAGEAD's Representation of Linguistic Knowledge

MAGEAD relates (bidirectionally) a lexeme and a set of feature-value pairs to a surface word form through a sequence of transformations. In a generation perspective, the features are translated to abstract morphemes which are then ordered and expressed as concrete morphemes. The concrete templatic morphemes are interdigitated and affixes added, and finally morphological and phonological rewrite rules are applied.

MAGEAD defines the lexeme to be a triple consisting of a root and a **morphological behavior class (MBC)**. The MBC is characterized by the part-of-speech and the inflectional paradigm. For verbs, the inflectional paradigm is closely identified with the pattern and the transitivity. MAGEAD uses a representation of the morphemes which is independent of the specific variant of Arabic (Standard or dialect). These morphemes are referred to as **abstract morphemes** (AMs). For instance, in our example, the MBC [mbc:verb-I-au-tr] maps the two feature-value pairs [asp:I] and [vox:act] to the two AMs: [PAT_IV:I] and [VOC_IV:I-au-act].

The AMs are then ordered into the surface-order of the corresponding concrete morphemes. The ordering of AMs is specified in a **context-free grammar (CFG)**. This particular CFG is non-recursive and compiles to an FSA; we use a CFG and its non-terminals only for descriptive clarity. The ordered AMs for our example look like this:

(2) [CONJ:w][PREP:l][SUBJPRE_IV:3MP]
[d][ς][w][PAT_IV:I][VOC_IV:I-au-act]
[SUBJSUF_IV:3MP_Sub][OBJ:3MS]

The AMs are then translated to their corresponding variant-specific **concrete morphemes (CMs)**:

(3) wa+ li+ y+ ⟨V12V3,dʕw,au⟩ +ū +hu

Next, the morphemic representation is obtained by interdigitating the templatic morphemes (root/vocalism/pattern):

(4) wa+ li+ y+ adʕuw +ū +hu

MAGEAD has two types of rules. **Morphophonemic/phonological rules** map from the morphemic representation (4) to the phonological and orthographic representations. This includes default rules which copy the root and vocalism to the phonological and orthographic tiers, and specialized rules such as the rules that handle the hollow and defective verbs (our example is a defective verb, it has a glide /*w*/ in its final radical). Note that a simple concatenation of the morphemes in (4) will result in */*waliyadʕuwūhu*/ which is a wrong surface (phonological) form. In our example, a morphophonemic rule mandates assimilating the sequence */uw/* with the suffix /+*ū*/. The phonological representation of our example is:

(5) wa+ li+ y+ adʕ +ū +hu

Orthographic rules rewrite only the orthographic representation. These include, for example, rules for using the Shadda (consonant doubling diacritic). In our example, the orthographic rule rewrites the verbal suffix long-vowel +ū as (ـُوا+) +*uwA* in a final position or as (ـُو+) +*uw* in medial position (i.e., when followed by a pronominal object as in our example). The orthographic representation is:

(6) wa+ li+ y+ adʕ +uw +hu

4.2 Implementation of MAGEAD in FSTs

MAGEAD follows Kiraz (Kiraz, 2000) in using a multi-tape analysis. The five tiers are used as follows: tier 1 (pattern and affixational morphemes), tier 2 (root), tier 3 (vocalism), tier 4 (phonological representation), and tier 5 (orthographic representation). In the generation direction, tiers 1 through 3 are always input tiers. Tier 4 is first an output tier, and subsequently an input tier. Tier 5 is always an output tier. All tiers are read or written at the same time, so that the rules of the multi-tier transducer are rules that scan the input tiers and, depending on the state, write to the output tier.

MAGEAD is implemented as a multi-tape finite state transducer layer on top of the AT&T two-tape finite state transducers (Mohri et al., 2000). Conversion from this higher layer (the **Morphtools format**) to the Lextools format (an NLP-oriented extension of the AT&T toolkit for finite-state machines (Sproat, 1995)) is done for different types of Lextools files such as rule files or context-free grammar files. A central concept here is that of the **multi-tier tokens** (MTT), which is a token which represents five tiers but which is compatible with Lextools. An MTT is a sequence $[T, R, V, P, O]$ where: *T* is a token from the pattern "template", *R* is a root radical, *V* is a vowel from the vocalism, *P* is a token from the phonological representation, and *O* is a letter from the orthographic representation. The first (or pattern) tier (*T*) is always required. The additional tiers can be left underspecified or empty (ε), which is both represented with the symbol 0. For example, the information in (3) is conceptually represented in the multi-tier system as follows (only the top three tiers are filled in since no processing has taken place yet):

In the implementation, each column becomes one MTT (i.e., a single symbol in the underlying FST), and the information in (7) is actually represented as follows:

(8) [w0000] [a0000] [+0000] [l0000] [i0000] [+0000] [y0000] [+0000] [V0a00] [1d000] [2ʕ000] [V0u00][3w000] [+0000] [ū0000] [+0000] [h0000] [u0000]

After applying phonological rules, the fourth (phonological) tier has been filled in in each MTT:

(9) [w00w0] [a00a0] [+0000] [l00l0] [i00i0] [+0000] [y00y0] [+0000] [V0aa0] [1d0d0] [2ʕ0ʕ0] [V0uu0][3w000] [+0000] [ū00ū0] [+0000] [h00h0] [u00u0]

Note that the last radical in the stem [3w000] did not map to the phonological layer due to the morphophonemic rules discussed in the previous section. In this fourth tier, this represents the phonological form /waliyadʕuūhu/. Orthographic rules are then applied which write symbols into the fifth tier and to modify them, ultimately yielding *waliyadʕuwhu*. Note that the fourth tier provides the (phonemic) pronunciation for the orthography in

(7)

Pattern Tier	w	a	+	l	i	+	y	+	V	1	2	V	3	+	ū	+	h	u
Radicals Tier	0	0	0	0	0	0	0	0	0	d	ς	0	w	0	0	0	0	0
Vocalism Tier	0	0	0	0	0	0	0	0	a	0	0	u	0	0	0	0	0	0
Phonological Tier	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Orthographic Tier	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

the fifth tier. The orthographic tier always has diacritics; it differs from the phonological tier in terms of spelling conventions relating to the Ta-Marbuta, the Alif Maqsura, and the Hamza forms. This does not mean that as an analyzer, the system always requires diacritized input: the input to the analyzer can be fully diacritized, partially diacritized, or undiacritized, since the operational system includes a step of hypothesizing diacritics if they are absent.

4.3 Lexicon

One of the main design goals of MAGEAD was to be able to operate without a lexicon or with only a partial lexicon. The motivation is that despite the similarities between dialects at the morphological and lexical levels, it is hard to build a complete lexicon for every dialect. The lexicon in MAGEAD operates as a filter that is not part of MAGEAD's FSTs. After the morphological analyzer generates all morphologically possible analyses, the lexicon removes those analyses that do not correspond to lexical entries. Therefore, running MAGEAD without a lexicon comes at the cost of over-generation. We created another version of MAGEAD that has the lexicon compiled into its FST, we call this version MAGEAD-LEX. We will discuss MAGEAD-LEX in more details in Section 7.

5 MAGEAD-EXPRESS

Similar to MAGEAD, MAGEAD-EXPRESS is a morphological analyzer and generator for Arabic and its dialects that also analyzes to or generates from a lexeme and a set of linguistic feature-value pairs. MAGEAD-EXPRESS is composed of two parts: the **linguistic resources** and the **the morphological engine**. MAGEAD-EXPRESS's linguistic database follows the general structure of BAMA (Buckwalter, 2004) and ALMORGEANA (Habash, 2007). The main difference is that MAGEAD-EXPRESS's databases are extracted automatically from MAGEAD's FSTs. This section will give an overview on the structure of the linguistic information. Section 6 will go over the extraction process in detail.

An Arabic word can be viewed as a concatenation of three regions: a prefix, a suffix and a stem; only the prefix and suffix regions can be null. MAGEAD-EXPRESS's database consists of three lexicons, one for each word-region and three **compatibility tables** (CTs): prefix-stem, stem-suffix and prefix-suffix CT. Prefix and suffix lexicon entries cover all possible concatenations of Arabic prefixes/proclitics and suffixes/enclitics respectively. Similarly, the stem entries cover all possible stems for each lexeme. Each entry, in any of the lexicons, is *minimally* composed of four fields: undiacritized surface form, morphological category, diacritized surface form and morphological feature-value pairs associated with the entry. In our example وَلِيَدعُوهُ *waliyadςuwhu* 'and that they invite him', the prefix, stem and suffix entries are, respectively:

(10) wly pre-10 diac:waliy asp:I per:3 gen:m num:p cnj:w prt:l

(11) dς stem-60 diac:adς 1:d 2:ς 3:w mbc:verb-I-au-tr asp:I vox:act

(12) wh suf-23 diac:uwhu asp:I mod:s per:3 gen:m num:p pro:3MS

The CTs specify which morphological categories are allowed to co-occur. In our example, pre-10, stem-60, and suf-23 have to be three-way compatible to generate our example verb. Because the CTs are built automatically in MAGEAD-EXPRESS, the name of a category is nothing but a unique identifier; unlike BAMA/ALMORGEANA's categories that have human-interpretable meanings.

MAGEAD-EXPRESS utilizes the morphological analysis/generation engine of ALMORGEANA as it complies with the two main general specifications of MAGEAD **analysis/generation** to/from **a lexeme and feature-value pairs**. In analysis, the word is segmented into all possible sets of prefix, stem and suffix strings. In a valid segmentation, the strings should exist in their corresponding lexicons and their categories should be compatible. Generation is similar to analysis but instead of matching on surface forms, the matching occur on the features. For a valid generation, the surface forms corresponding to the compatible sets of features of each word part are then concatenated to form the final word form.

6 Extracting MAGEAD-EXPRESS Tables from MAGEAD FSTs

In this section, we present how we extract the linguistic information from MAGEAD's FSTs and present it in a tabular format compatible with MAGEAD-EXPRESS as descried in section 5. The extraction process takes place in an incremental fashion where more information is added to the initial tables in each step.

Creating the Initial AM Tables: MAGEAD's context-free grammar (CFG) automaton (introduced in Section 4.1) encodes all the possible combinations of abstract morphemes (AMs) that are compatible. We start by listing all possible AM combinations. For instance, the AM combination responsible for the analysis or generation of our example, وَلِيَدعُوهُ *waliyadςuwhu* 'and that they invite him', is as in (2). We separate the prefixes, stems and suffixes into three different tables by composing the list with appropriate FST filters that delete un-needed AMs. We also restrict the entries in the stem-AM table to the set of lexemes that exists in MAGEAD's lexicon. Each AM subsequence receives a compatibility category, which is computed as follows. First, we populate three compatibility tables (CTs): prefix-stem, stem-suffix and prefix-suffix that specify compatibility of the AM subsequences. We cluster the prefixes into sets whose member are all compatible with the exact same sets of stems and suffixes; the set is assigned an automatically generated compatibility category name. The same happens to stems and suffixes. Finally, we augment the entries of the prefix-AM, stem-AM and suffix-AM tables with their compatibility categories.

For example, the AM sequence in (2) would contribute the following three AM subsequences (paired with their compatibility categories) to the prefix-AM, stem-AM and suffix-AM tables, respectively:

(13) pre-AM-1 [CONJ:w][PREP:l]
[SUBJPRE_IV:3MP]

(14) stem-AM-1 [1d][2ς][3w][PAT_IV:I]
[VOC_IV:I-au-act]

(15) suf-AM-14 [SUBJSUF_IV:3MP_Sub]
[OBJ:3MS]

Creating the Morphemic Tables: Each AM subsequence in the prefix-AM, stem-AM and suffix-AM tables is composed with the abstract morpheme-to-concrete morpheme transducer that is responsible for translating the AM to their equivalent CM represented in multi-tier tokens (MTTs). Also, the AMs are composed with the abstract morpheme-to-feature transducer that maps the AMs to their linguistic features. This results in a new set of tables that also includes the morphemic representation and the corresponding set of the linguistic feature-value pairs. There is no change in categories in this phase. Each entry in any of the morphemic tables has four columns ordered as: CM, compatibility category, AM, feature-value pairs. Examples of the tables are as follows for prefix, stem and suffix, respectively:

(16) [w0000][a0000][+0000][l0000][i0000][+0000]
[y0000][+0000] pre-AM-1 [CONJ:w]
[PREP:l][SUBJPRE_IV:3MP] [asp:I]
[per:3][gen:m][num:p][cnj:w][prt:l]

(17) [V0a00][1d000][2ς000][V0u00][3w000]
stem-AM-1 [1d][2ς][3w][PAT_IV:I]
[VOC_IV:I-au-act] [1:d][2:E][3:w]
[mbc:verb-I-au-tr][asp:I][vox:act]

(18) [+0000][ū0000][+0000][h0000][u0000]
suf-AM-14 [SUBJSUF_IV:3MP_Sub]
[OBJ:3MS] [asp:I][mod:s][per:3][gen:m]
[num:p][pro:3MS]

Creating the Surface Form Tables: The CM of prefixes, stems and suffixes cannot be converted separately to their surface form because there are morphemic, phonological and orthographic rules that target interactions among them. At this stage we use the AM categories to form all possible morphemic representations in our tables. So for our example, the morphemic representation is (8). The morphemic representations are then composed with MAGEAD's generation FST where the morphemic, phonological and orthographic rules are applied. We also keep track of the word parts to be able to divide the surface form later to prefix, stem and suffix by using +'s to mark the boundaries. The result of this operations is as follows:

(19) waliy+ adς +uwhu

Applying the rules creates new surface forms that require new categories. For example, as discussed earlier, the verbal suffix long-vowel +ū is rewritten as (ـُوا+) *+uwA* in a final position or as (ـُو+) *+uw* in a medial position. These two new surface forms need two new categories because they are no longer compatible with everything +ū was compatible with. The surface forms and the new categories are then added to the entries in the morphemic tables and a new set of surface-form tables is created (prefix, stem, and suffix, respectively):

(20) waliy pre-10 [w0000][a0000][+0000][l0000] [i0000][+0000][y0000][+0000] pre-AM-1 [CONJ:w][PREP:l][SUBJPRE_IV:3MP] [asp:I][per:3][gen:m][num:p][cnj:w][prt:l]

(21) adς stem-60 [V0a00][1d000][2ς000] [V0u00][3w000] stem-AM-1 [1d][2ς] [3w] [PAT_IV:I][VOC_IV:I-au-act] [1:d] [2:E][3:w][mbc:verb-I-au-tr][asp:I][vox:act]

(22) uwhu suf-23 [+0000][ū0000][+0000] [h0000][u0000] suf-AM-14 [SUBJ-SUF_IV:3MP_Sub][OBJ:3MS] [asp:I] [mod:s][per:3][gen:m][num:p][pro:3MS]

In a final step, we put these tables in the appropriate format for ALMORGEANA as described in section 5.

7 Evaluation

MAGEAD-EXPRESS uses MAGEAD's linguistic resources; therefore its lexical coverage should be identical to that of MAGEAD. We do not evaluate MAGEAD's coverage here; for more information about MSA and Levantine verb coverage, see (Habash and Rambow, 2006) and for MSA nouns, see (Altantawy et al., 2010). In this section, we evaluate the conversion of MAGEAD's linguistic resources for MSA Verbs from their FST representation to the tabular representation used by MAGEAD-EXPRESS. We also report on time performance and memory usage of MAGEAD-EXPRESS versus MAGEAD.

Both MAGEAD and MAGEAD-EXPRESS are bidirectional systems. It is sufficient to evaluate them either in analysis or generation mode. For the purpose of this evaluation, we opted to do the generation mode only. The goal of this evaluation is to ensure that given any lexeme from MAGEAD's lexicon paired with a plausible set of linguistic feature-value pairs, both MAGEAD and MAGEAD-EXPRESS will generate the same surface forms.

MAGEAD's verb lexicon contains 8,960 lexemes, each of which has either 1,092 inflected surface forms if the lexeme is for an intransitive verb or 14,196 if transitive (accepts pronominal object clitics). Since it is too time-consuming to test all the 8,960 lexemes, we create a representative sample of lexemes to serve as an evaluation dataset. We cluster MAGEAD's lexicon into 611 lexeme groups. Each group represents an MBC with a particular root type that triggers a particular set of rewrite rules. Each lexeme group is assigned an **iconic lexeme** (IL) that represents the group. The 611 ILs (240 intransitive and 371 transitive) are used as our evaluation dataset. Each IL in the evaluation dataset is paired with all possible combinations of feature-value pairs and then fed to both MAGEAD and MAGEAD-EXPRESS. The surface forms generated by both systems were identical in all cases. This validates the correction of our conversion process.

We now evaluate the time and memory requirements of MAGEAD-EXPRESS against MAGEAD. For the sake of this evaluation, we also created another version of MAGEAD that has the lexicon compiled into its FST; we call this version MAGEAD-LEX. Table 7 compares MAGEAD-EXPRESS to both MAGEAD-LEX and MAGEAD. MAGEAD-LEX is restricted by the lexicon and thus can not operate on any lexemes that are not in the lexicon (unlike MAGEAD but like MAGEAD-EXPRESS). MAGEAD is the only system among these three that can run without a lexicon. MAGEAD-EXPRESS is the only system among the three that allows easy access to intermediate representations such as those appearing in (20)-(22), i.e., AMs and CMs.

	MAGEAD-EXPRESS	MAGEAD-LEX	MAGEAD
Embedded lexicon	Yes	Yes	No
Can run without a lexicon	No	No	Yes
Intermediate representations in output	Yes	No	No
Time to build	30mins (MAGEAD) + 48hrs	30mins (MAGEAD) + 30mins	30mins
Time to analyze 10K verbs (batches of 1,000)	68 secs	100 secs	3,985 secs
Time to analyze 1 verb	3.5 secs	4.5 secs	2.1 secs
Time to analyze 1 verb online (client-server)	0.00679 secs	No	No
Size of machines	7MB	388MB	Not Composable 13M⊙14M⊙583K

Table 1: Comparing MAGEAD-EXPRESS, MAGEAD-LEX, and MAGEAD. All reported times are CPU seconds.

In terms of time to build, MAGEAD takes 30 mins to compile (MAGEAD-LEX takes an hour), and then MAGEAD-EXPRESS takes around 48 hours to extract its tables from MAGEAD's FSTs.

As for time performance, we created a list of 10,000 verbs collected randomly from the list of surface forms generated in the extraction evaluation. We measure the time MAGEAD-EXPRESS takes to finish analyzing the 10,000 verbs (in ten batches of 1,000 verbs each) against the time taken by MAGEAD and MAGEAD-LEX. As Table 7 shows, MAGEAD-EXPRESS is 1.5 times as fast as MAGEAD-LEX and 58 times faster than MAGEAD. We next compute the average speed of analyzing one verb at a time offline (using all 10,000 verbs). In this scenario, MAGEAD is the fastest due to the overhead time (3.4 secs) that MAGEAD-EXPRESS needs to load its tables into memory. Of course, MAGEAD-EXPRESS is the only system among the three systems that can operate in a client-server setup where it loads its lexicons once into memory. In such setup, MAGEAD-EXPRESS is more than 300 times faster than MAGEAD.

As for memory requirements, MAGEAD-EXPRESS requires about 7MB to store its tables into memory. When the tables get loaded in memory, MAGEAD-EXPRESS does not require additional resources. On the other hand, MAGEAD consists of three FSTs that are composed with the input in an online fashion, see Table 7 for sizes. In fact, we could not compose any two of MAGEAD's three FSTs into a bigger FST on our 64GB memory machine. However, MAGEAD-LEX is much smaller than MAGEAD because it is only restricted to the lexicon. MAGEAD-LEX's FSTs are composed into one big FST of size 388MB. The amount of memory needed by MAGEAD depends mainly on the size of the input. For 1 verb to be analyzed, the input is composed with MAGEAD's three FSTs and the result is an FST of size 10KB, and for a batch of 10, 100, and 1,000 verbs the resulting FST is of size 113KB, 1.7MB, and 13MB, respectively.

8 Conclusion

In this paper, we introduced MAGEAD-EXPRESS, a lexicon-based morphological analyzer and generator for Arabic and its dialects. MAGEAD-EXPRESS extracts its linguistic knowledge automatically from MAGEAD. As a result, MAGEAD-EXPRESS still benefits from the level of abstraction with which the linguistic information is encoded in MAGEAD's FSTs, while being much faster than MAGEAD. Both systems are bidirectional and analyze to and generate from a lexeme and a set of linguistic feature-value pairs. MAGEAD's main advantage over MAGEAD-EXPRESS, is its ability to work without a lexicon or with a partial lexicon. In ongoing work, we are studying how we can combine the two system to build a more extensive system. The main idea is to port MAGEAD as a back-off mechanism to deal with the out-of-vocabulary words that are not in the lexicon. Although we only demonstrate the FST-table conversion idea on Arabic, we believe it is applicable to other languages with comparable benefits (depending on the language's morphological complexity).

References

Imad Al-Sughaiyer and Ibrahim Al-Kharashi. 2004. Arabic Morphological Analysis Techniques: A Comprehensive Survey. *Journal of the American Society for Information Science and Technology*, 55(3):189–213.

Muhammed Aljlayl and Ophir Frieder. 2002. On Arabic Search: Improving the Retrieval Effectiveness via a Light Stemming Approach. In *Proceedings of ACM Eleventh Conference on Information and Knowledge Management, Mclean, VA*, pages 340–347.

Mohamed Altantawy, Nizar Habash, Owen Rambow, and Ibrahim Saleh. 2010. Morphological analysis and generation of arabic nouns: A morphemic functional approach. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC. Valletta, Malta*.

Saba Amsalu and Dafydd Gibbon. 2005. Finite state morphology of Amharic. In *International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 47–51, Borovets.

Kenneth Beesley and Lauri Karttunen. 2000. Finite-state non-concatenative morphotactics. In *acl00*, Hong Kong, China.

Kenneth Beesley, Tim Buckwalter, and Stuart Newton. 1989. Two-level finite-state analysis of Arabic morphology. In *Proceedings of the Seminar on Bilingual Computing in Arabic and English*, page n.p.

Kenneth Beesley. 1998. Arabic morphology using only finite-state operations. In M. Rosner, editor, *Proceedings of the Workshop on Computational Approaches to Semitic Languages*, pages 50–7, Montereal.

Tim Buckwalter. 2004. Buckwalter arabic morphological analyzer version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Michael Gasser. 2009. Semitic morphological analysis and generation using finite state transducers with feature structures. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 309–317, Athens, Greece.

Nizar Habash and Owen Rambow. 2006. MAGEAD: A Morphological Analyzer and Generator for the Arabic Dialects. In *Proceedings of the Conference of the Association for Computational Linguistics*, Sydney, Australia.

Nizar Habash, Owen Rambow, and George Kiraz. 2005. Morphological Analysis and Generation for Arabic Dialects. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, Ann Arbor, Michigan.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash. 2007. Arabic Morphological Representations for Machine Translation. In Antal van den Bosch and Abdelhadi Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Kluwer/Springer.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Laura Kataja and Kimmo Koskenniemi. 1988. Finite state description of Semitic morphology. In *COLING-88: Papers Presented to the 12th International Conference on Computational Linguistics*, volume 1, pages 313–15.

Martin Kay. 1987. Nonconcatenative Finite-State Morphology. In *Proceedings of the Conference of the European Chapter of ACL (EACL-87)*, Copenhagen, Denmark.

Ahmed El Kholy and Nizar Habash. 2010. Orthographic and Morphological Processing for English-Arabic Statistical Machine Translation. In *Actes de Traitement Automatique des Langues Naturelles (TALN)*, Montreal, Canada.

George Kiraz. 2000. Multi-tiered nonlinear morphology using multi-tape finite automata: A case study on Syriac and Arabic. *Computational Linguistics*, 26(1):77–105.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231:17–32, January.

Otakar Smrž. 2007. *Functional Arabic Morphology. Formal System and Implementation*. Ph.D. thesis, Charles University in Prague, Prague, Czech Republic.

Richard Sproat. 1995. Lextools: Tools for finite-state linguistic analysis. Technical Report 11522-951108-10TM, Bell Laboratories.

Shlomo Yona and Shuly Wintner. 2005. A finite-state morphological grammar of Hebrew. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 9–16, Ann Arbor, Michigan.